Implementation of the Model “Simulating Collective Misbelief” into a NetLogo environment

Bachelor Thesis

for the achievement of a Bachelor of Science degree
Course of studies: Information Management

submitted by:
Kai Manuel Hemmerich
Matriculation number: 205210220
E-Mail: khemmerich@uni-koblenz.de

Due Date: 24.02.2012

University of Koblenz-Landau
Department 4: Informatics
Institute for IS Research

Supervising Professor:
Prof. Dr. Klaus G. Troitzsch
Second Supervisor:
Dr. rer. nat. Michael Möhring

Koblenz, Februar 2012

Explanation

Ich versichere,
dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

I certify that I have written this thesis independently and that no other than the mentioned sources and aids where used.

I agree with the publishing in the library. I agree with the publishing of this thesis on the inter-net.

Table of content

Table of figures and tables

1 Introduction

Jim Doran developed a model called ‘Simulating Collective Misbelief’ (Doran 1998) to discover the impact of misbelief on an artificial society.

Misbelief respectively belief are phenomena that can be found through all times, cultures and societies. In medicine, belief is an important factor for the healing process. A placebo for example is an imitation of a real medicament without the medical substances and their specific effects (Hehlmann 1965). By treatment with placebos positive effects can be observed on the patient. Consequently psychological factors (one of them is belief in the effectiveness of the medicament) can have a significant impact on medical recovery.

Accordingly one could say that misbelief has a negative impact. In medicine this is called nocebo-effect (Olshansky 2007). The fear of the patient about side effects results in a negative medical effect on the patient. Another example of a similar negative impact is the so called mass suicide of Jonestown (Bundesverband 2008). A group of people (mis-) believed in their leader Jim Jones (who felt persecuted). He told his followers that there was an imminent attack by a mercenary group and the only way out would be a collective suicide. More than 1,000 people died by committing mass suicide.

In contrary to this, Jim Doran provides an example (Doran 1998) of a positive impact of misbelief. He cites a group of people whose members (incorrectly) believe, that a certain stream of water is holy. Based on that, it is forbidden to take water out of that stream. At first, this is a disadvantage for the group because they have to find water elsewhere. But the (mis-) belief may also be a benefit if the stream is dangerously contaminated. In the eyes of the group their belief was correct but from an observer’s[1] point of view it was a misbelief because the survival of the group did not base on the holiness of the stream.

On the observer’s side the belief was replaced by knowledge of all circumstances without any limitations. Further the belief of each individual influences its behavior. In the example above the group did not take any water out of the stream due to their belief. Experiments in the area of misbelief would cause a danger (humans could die or injured) for the participants. Out of these possible impacts of an experiment it seems to be suitable to work with computer simulations for evaluating the impact and simulating collective misbelief.

[1] The observer is a person who has knowledge about all circumstances of the situation. In the NetLogo simulation an observer can directly intervene into a running simulation and can change variables or parameters.

Doran’s basic statement is that agents beliefs in a multiple agent system are often partial wrong or inconsistent. This misbelief not necessarily damages the system at a whole. He focuses on a positive impact of misbelief. This leads to the research question:

“Is it right that misbelief has a positive impact to a society?”

1.1 Target of the thesis

The goal of this Bachelor thesis is to implement and evaluate the “Simulating of Collective Misbelief”-model into the NetLogo programming language. Therefore, the model requirements have to be specified and implemented into the NetLogo environment. Further tool-related requirements have to be specified to enable the model to work in NetLogo. After implementation several simulations will be conducted to answer the research question stated above.

1.2 Structure of the thesis

The following Chapter 2 will introduce the applied research method. Chapter 3 discusses the theoretical background of the scientific area of collective misbelief. It will follow up with a description of the two scenarios of Doran’s model (Doran 1998) including the used components and a set of behavioral rules within the model. After the implementation part (Chapter 4) the analysis chapter (Chapter 5) discusses the simulation results in relation to Doran’s previous analysis and conclusions. This work concludes with a critical appraisal and a future view on how Doran’s model or the corresponding simulation of the model could be extended or enriched for further research.

2 Research Method

Wilde and Hess mapped research methods of the information system research area into a method profile which is shown in figure 2-1 (Wilde and Hess 2007). They analyzed 300 research papers of the Journal “Wirtschaftsinformatik” from 1996 to 2006. Within that profile the identified research methods are classified according to a formalization scale (with a qualitative and a quantitative expression) and a paradigm scale (with behavioral and constructional expressions). This thesis is placed in the field of simulation, which is a quantitative and constructive method. A concrete model must be implemented using computer simulation, executed and finally evaluated. Rules for proceeding, modeling and evaluation are typical components for design-oriented methods (Wilde and Hess 2006).

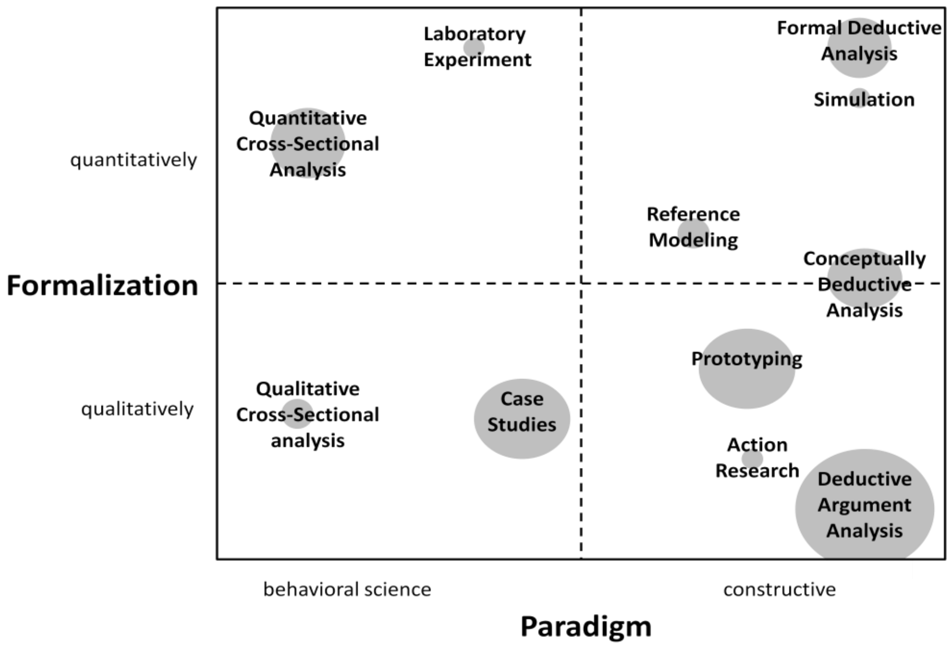


Figure 2-1: Empirical based method-profile of information systems research, based on (Wilde and Hess 2007)

Furthermore, this thesis can be linked to the design-science approach. Design science belongs to a problem-solving paradigm. New and innovative artifacts are created and evaluated to solve identified problems (Hevner, et al. 2004, 77). The created artifact in this work enables researchers to evaluate Doran’s given theory in an efficient way and belongs therefore to the design science approach (Hevner, et al. 2004, 81 ff). In this thesis the artifact can be addressed to the executable NetLogo source-code as a software-component. Hevner developed seven guidelines for Design Science Research these were applied in this thesis:

1. Design as an Artifact

Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.

The first guideline addresses the artifact[2] as the core component of design oriented work. According to this guideline Design Science Research must produce a viable artifact. This artifact could be some form of construct[3], a model[4], a method[5] or an instantiation[6] (Hevner, et al. 2004). The current thesis produces a NetLogo simulation that serves as an instantiation of a given problem and an evaluation tool for a given proposition based on the identified problem.

2. Problem Relevance

The objective of design-science research is to develop technology-based solutions to important and relevant problems.

The second guideline emphasizes the relevance of the problem and points out that it is important to develop a technology-based solution. The problem relevance is given by Doran's proposition that the inconsistent and wrong belief of agents in a multiple agent system not necessarily damage the system as a whole. The simulation in this thesis is developed for NetLogo, a computer-based simulation tool. Thereby, both requirements of guideline two are fulfilled.

3. Design Evaluation

The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.

Guideline three points the design evaluation method to proof the utility, quality and efficiency of the artifact. Hevner defines a complete and effective artifact.

"… when it satisfies the requirements and constrains of the problem…"

The simulation program includes all requirements and constraints described by Jim Doran (Simulating Collective Missbelief 1998). It is a simulating experimental evaluation method and meets the requirements of guideline three.

[2] An artifact could be a piece of software, a whole program or another efficient IT component.

[3] Vocabulary and symbols

[4] Abstractions and representations

[5] Algorithms and practices

[6] Implemented and prototype systems

4. Research Contributions

Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.

Guideline four mentions the research contributions on several levels (artifact, foundation, methodology). The main contribution of this work is the artifact itself. The NetLogo simulation offers researchers many options to change variables with the given settings and evaluate the results on a state-of-the-art technology basis. This fits Hevners first explanation that the designed artifact itself could succeed this guideline (Hevner, et al. 2004).

5. Research Rigor

Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.

Guideline five calls for the application of rigorous methods in both the construction and evaluation of the design artifact. One task of this thesis is the implementation part of an existing model into a NetLogo simulation environment. The implementation of the model followed a "code and fix"[7] approach. The code is developed in several iterative process-steps, is tested and is recoded. After the implementation the model is tested, evaluated[8] and linked to Doran's results.

6. Design as a Search Process

The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.

Guideline six describes design as a search process. The focus of this thesis is to implement and check an existing model. Therefore the process is defined by identifying the needed components, a behavioral rule set, the environmental constraints, the changing variables and validates the model results.

7. Communication of Research

Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Guideline seven addresses the communication of research, the importance of the work for the academic community and the practitioner's community. This thesis implements Doran's model into a state-of-the-art simulation environment and enables researches to discover further knowledge by detailed variations of variable settings. Additionally several issues are raised for further research.

All these guidelines were addressed in this work and can be found in a more detailed description in the following chapters.

[7] Code and fix approach is a software development method for small software-projects. (Dooley 2011)

[8] The found variable-combination is validated by 40 simulation runs

3 Theoretical background

This chapter presents theoretical background information that influences the implementation of the model or is needed for a better understanding of the model.

3.1 Computer Simulation

Computer simulation is a technique that attempts to simulate a real world element in a computer program (McHaney 1991). It can be especially helpful in cases where the experiment bases on complex nonlinear models or e.g.is treated as unethical[9] (Troitzsch and Gilbert 2005). The computer simulation is not a substitute for analytical or empirical research. Computer simulation can rather be seen as a tool for researchers to model the “real world” in a virtual environment.

> *“We define simulation as a method for using computer software to model the operation of the “real-world” process, systems, or events”* (Davis, Eisenhardt und Bingham 2007).

Davis, Eisenhardt and Bingham (2007) further identify strengths and weaknesses of simulation. A distinct strength of a simulation is the strong internal validity that is accomplished by precise specifications of the construct and a rigor defined theoretical logic. Nevertheless, this strength can only be fostered if the underlying theoretical model is sound. Empirical Research often provides a poor defined theoretical logic with weak specifications due to incorrect assumptions in the model.

Davis, Eisenhardt and Bingham (2007) identify an additional strength of simulation in creating a computational laboratory in which researchers can change variables, add new features or change the model in a controlled environment. Harrison (2007) mentions that new knowledge is not only created by interpreting the simulation results but that model creation and process definition also informs the body of knowledge. Simulations often start with a simplified description of the model environment and a simple rule set of for the agents-programming. During the setup of the model these simplifications have to be clarified further which in turn reflect on the overall model design and its underlying assumptions. On the other hand in social simulation the simplification and the simple rule set of the agents can lead to a complex behavior of the agents-population (Troitzsch and Gilbert 2005).

This thesis is based on a multi-agent-simulation approach for which an appropriate software platform had to be found. The number of software-platforms and the quality of these platforms increase rapidly. Railsback et al (2006) compare 4 simulation platforms (“MASON”, “NetLogo”,

[9] That means that the acquisition of knowledge would not equal the “use of resources”

“Repast” and “Java/Objective-C version of Swarm”). For evaluation purpose, they implemented several models with raising complexity in all platforms and focused assessment criteria like “ease of use”, “execution speed” or “Quality of documentation”. They concluded that

> *“NetLogo is the highest-level platform, providing a simple yet powerful programming language, built-in graphical interfaces, and comprehensive documentation.” (Railsback, Lytinen and Jackson 2006)*

“MASON” is designed to reach a high execution speed, which is no high priority in NetLogo. They remaining platforms have their individual benefits (e.g. focusing simulation speed “MASON” or combining advantages from two or using Java libraries for standardized software “Repast) that make them useful in this field of research.

For this thesis the NetLogo platform is chosen because it contains many functions, which simplify the programming. A further focus is the “ease of use” for the modeler who can change different parameters of the model in a simple and intuitive way.

3.2 NetLogo

NetLogo (Wilensky 1999) is a multi-agent programming language including a modeling environment. It was developed by Uri Wilensky at the Northwestern University's Center for Connected Learning and Computer-Based Modeling and was published as freeware in 1999. In general, the tool consists of three tabs with differing functions.

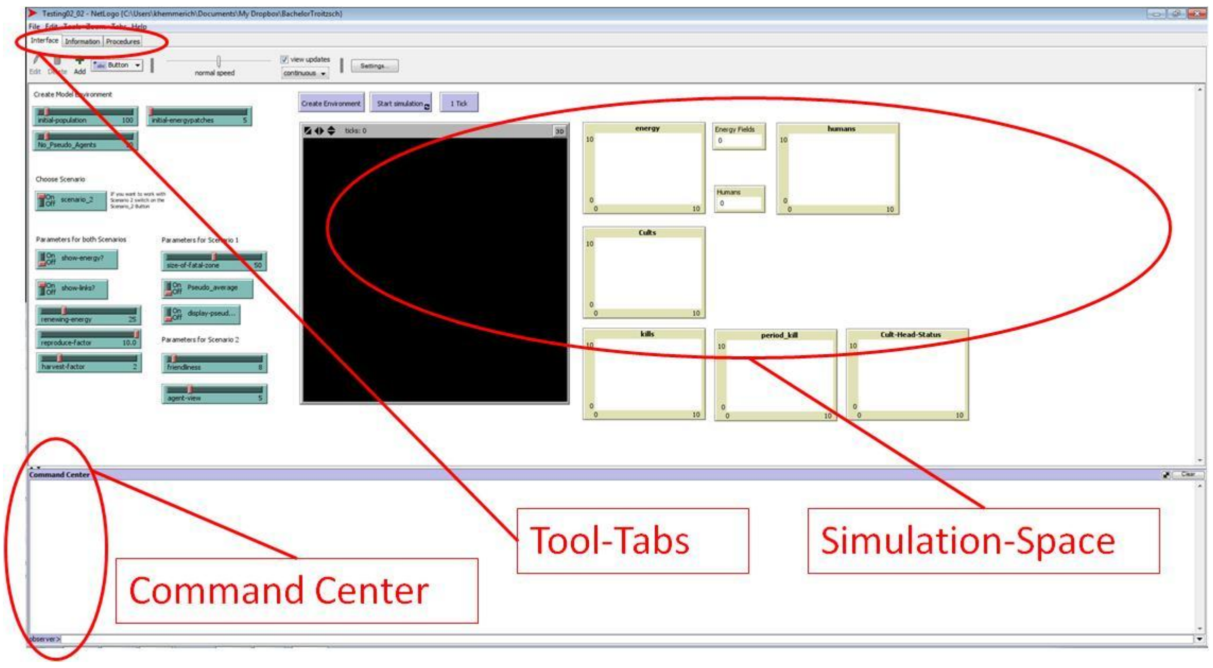


Figure 3-1: NetLogo-screenshot (own screen-shot)

The first tab is the interface-view. The model is visualized in this view. NetLogo works with an environment consisting of patches and agents (implemented as turtles). Both behave according to implemented model-rules. The modeler takes over the role of the observer who can add other elements (e.g. monitors, plots, buttons, slider, switches…) in the simulation-space[10]. These elements can be used to influence progress of the model. Within the "command center"[11] the observer can interact with the model and can give instructions to "agents" or "fields" in general, to a predefined group of them or to one single "agent" or "field". The observer can also trigger variables of the current model.

The second tab is the information-view. Here all needed information about the model, the methods, how to use the model and special things to be noticed can be written down. This view is just for information purposes and does not influence the model in any case.

The third tab is the procedures-view. Here, the source code of the model is located. The NetLogo programming language has its own syntax that can be checked at the online dictionary[12] or via a syntax-check-button.

3.3 Agents and artificial societies

Doran (1998) describes an agent as a "computational mechanism" that is situated in an environment. An agent behavior is influenced by its current internal state and its environment. The actions taken by an agent in turn influence the environment. Troitzsch and Gilbert (2005) note that there is no common definition for the entity of an agent.

Different classes of agents can be found in literature. A simple type of agent is the often called "reflex agent" (Russell and Norvig 1995). This type of agent follows a "condition-action-rules"[13] approach. If a special condition occurs, the agent executes a defined action.

A set of individual agents in a computational environment may interact with each other by sending messages (just information or request for information). Such multiple agent systems in a virtual environment are sometimes referred to as artificial society (Gilbert and Conte 1995)

An extension like an individual agent-memory that is updated from time to time leads to another type of agent which may be called "extended-agent" (Doran 1998). With such a type of agent memory the "belief" of an agent may be expressed that could influence its behavior.

[10] The area where the model is visualized and the observer can follow the progress of the model.

[11] The area where the observer can enter commands for the simulation and gets direct response.

[12] http://ccl.northwestern.edu/netlogo/docs/, last call: 05.01.2012

[13]Condition-action-rules are described as an if-then-action rule set (Russell and Norvig 1995).

3.4 Belief misbelief or knowledge

In a simulation an agent has to base its actions on information stemming from the environment as well as from their internal / individual memory information. Some of this information may be incorrect (e.g. no correct or missing update of the agent's information at a given point in time). Consequently agents have true information "knowledge" and potentially erroneous information "belief". This potentially wrong information may lead to actions the agent would not choose based on full or correct knowledge. Doran (1998) limits the description of belief to an abstract level while ignoring the matter of degrees of belief or a more detailed notation of belief. Therefore the abstract level is also sufficient for this thesis.

If a group of agents come to share the same "belief" it is called "collective belief".

From an "all knowing" observer point of view the information of an agent can be evaluated based on the information's veracity. The individual information of an agent represents its belief and if this belief is not true in the environment from the observer's point of view it is called misbelief (Doran 1998).

Consequently if a group of agents come to share the same misbelief it can be called "collective misbelief".

For the model "Simulating Collective Misbelief" Doran (1998) defines two types of misbelief in terms of "existence error" and "category error".

Existence error is the case if an agent believes in other agents who do not really exist. If an agent treats a resource-agent like a *human-agent,* this can be called **category error**.

3.5 Communication

The science of semiotics is divided into a syntactic[14], semantic[15] and pragmatic[16] view. Usually not every string of allowed signs (syntax) has an exactly one or equivalent sense (semantic). For a computer simulation on an abstract level it can be said that a given rule set is followed exactly and an agent will always do the same actions given the same inputs within the same circumstances. Therefore the syntactical rules have their equivalent one semantic sense.

[14] A rule set for correct expression and allowed signs for usage.

[15] The referencing of single signs to a sense.

[16] Is the transformation of the semantic into an operational behavior.

A lot of agents interact with each other in a multi agent system. The stimulus-response model is a simplification of the communication process consisting of three elements (Merten 1999):

- Communicator (source, author, sender, speaker …)
- Stimulus (information, speech, message, text …)
- Recipient (listener, target, audience, decoder …)

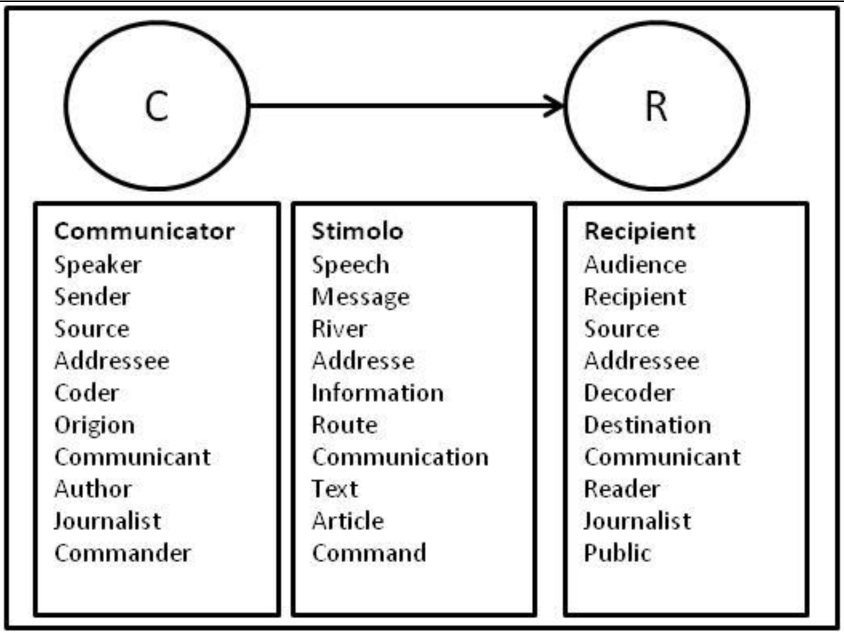


Figure 3-2: Stimulus-response model adapted from (Merten 1999)

The communicator (agent x) sends a stimulus (information) to the recipient (agent y) and the recipient respond to that stimulus. Figure 3-2 shows that the information transport is unidirectional from the communicator to the recipient. This kind of communication process can be found in Doran’s model when one agent hands over his information to another friend-agent within its view. The friend-agent is a kind of black box because it does not necessarily react to information transactions. A possible extension to that model is the option of agent y to answer (also mentioned by Merten (1999)). This kind of a bidirectional communication is used in the implementation when an agent asks a resource if it is the closest agent to the resource. Both communications will be explained in more detail in the following chapters.

4 Implementation into the NetLogo environment

Jim Doran developed a model to investigate the phenomena on collective misbelief in artificial society. Within this work the model will be implemented in a NetLogo environment. The thesis is based on the article “Simulating Collective Misbelief” by Jim Doran (1998) who developed two scenarios to investigate two kinds of (mis-) believes.

4.1 Description of both scenarios

Figure 4-1 show the general environment for scenario 1 with the needed components field (white squares), energy field (colored squares), fatal zone (red circle), human (blue human-shape) and *pseudo-agent* (yellow sad face).

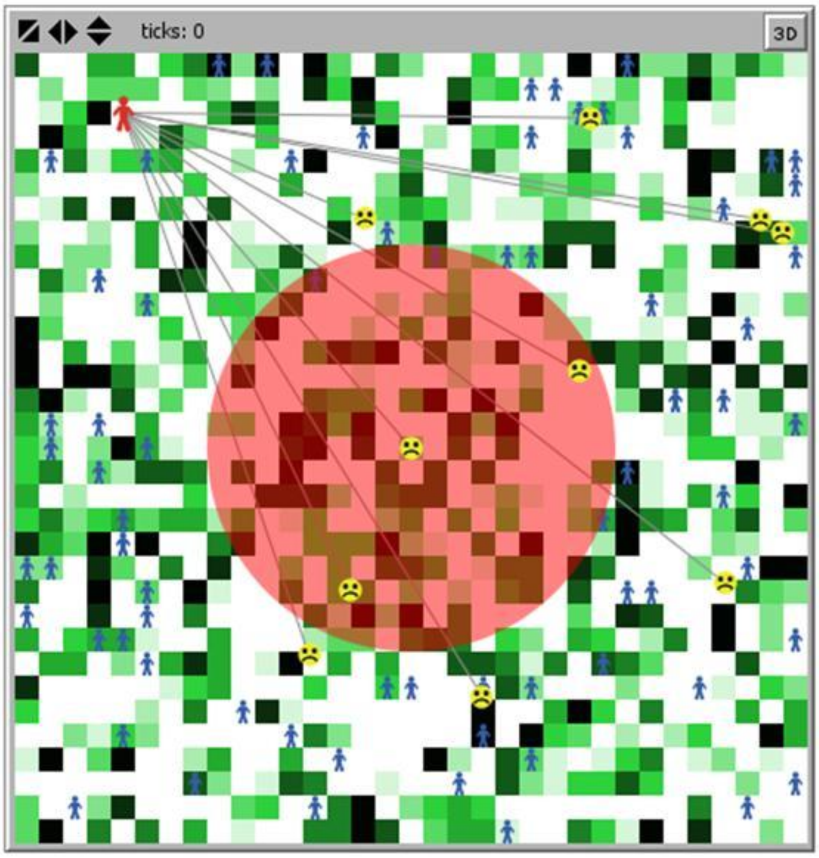


Figure 4-1: Example environment with related pseudo-agents of one human-agent (own screenshot)

Scenario 1[17] investigates the so-called existence error (see 3.4). A population (consisting of agents representing humans) “lives” on distinct fields in an environment. The second type of

[17] Equals Experiment I: The Impact Of Pseudo-Agents according to Doran (1998)

fields is the “energy field” where the humans can restore their energy[18]. Both “humans” and “energy fields” are randomly distributed at the start of each simulation. Each human loses energy by its movements during the simulation. If the energy level of a human is zero or lower the human “dies” and is removed from the environment. Further, the humans believe in a “fatal zone” where they immediately die by entry. This zone covers by a cycle (with the diameter equal to the 50%[19] wide of the environment) less than a quarter of the given environment. Additionally, each human believes in a small set of so-called *pseudo-agents*[20]. These *pseudo-agents* do not, in fact, exist in the environment and represent the “existence error”. *Pseudo-agents* stands for the (mis-) belief of each *human* because *humans* may pass up a good opportunity to harvest an energy patch due to the (mis-) believe in a *pseudo-agent*. A *human* is aware of the current location of all other *humans*, all resources and of its own *pseudo-agents*.

In general the simulation is divided into ticks that represent time units. Within each time unit an agent will take some fix actions. Firstly, a *human* harvests the energy-field at its current location and secondly, a small random movement takes place. *Humans* can additionally move towards the nearest resource under defined circumstances and within a set of rules and restrictions. Further *humans* are able to harmonize[17] their *pseudo-agents* and reproduce themselves. With these two processes Doran emphasized (1998) that there is a probability that a particular pattern of (mis-) beliefs may spread through the population.

In scenario 2 Doran describes another type of misbelief, the so-called category error. Due to this error, a human can comes to the belief that a resource-agent is of the type “human”. This (mis-) belief enables the *human* to take the *resources-agent* into its friend list. This error influences the future behavior of the *human* to the *resource-agent*. The friend list is another change in scenario 2 as well as the fact that *humans* can attack each other. In scenario 2 *humans* can add other *humans* (as well as resources) into their friend list. One *human* can attack another *human* if it is not in its friend list and if both have no other third *human* or resource as a common friend. *Humans* have a limited “agent-view” and pass their information about the location of resources to friends within their view. This “information-transport” is one-way, that means resources e.g. do not react to that information. Other *humans* receive the information and add them to their own information pool. However, they do not return their information in exchange. A cult is build around a resource or a human if they are in the friend list of other *humans*.

[18] All components, rule-set, restrictions and activities are explained later on

[19] The percentage-value can be changes by the observer.

[20] Pseudo-agent cannot move or consume the resources within the environment.

4.2 Used Elements

In this section all elements are introduced that are used in the simulation. Some are related to rules that are explained in other chapters and others are used out program requirements.

Environment

The environment of the simulation consists of a two dimensional square of 33x33 fields. Doran describes that

> *"…agents may leave the square as the experiment proceeds…"* (Doran 1998).

Within NetLogo the environment can be configured as an "open world"[21] or as a "closed world"[22]. As Doran did not define what happens if a *human* leaves the square (*human* dies, *human* is displayed on the opposite side of the square or something else) the world is configured as a closed world. *Humans* do focus on resource fields and move towards them.

Human

Humans are agents in the virtual world and map the society. In Scenario 1 each *human* has a random initial location, an initial energy level between 0 and 100 and believes in 10 *pseudo-agents*[23] (*pseudo-agents* will be explained in the following sections). During the simulation *humans* lose energy. They lose one unit of energy for each tick and units (in the amount of the round distance) for each movement towards a resource. A *human* has to move one field into a random direction but he can additionally move towards a resource if it is the closest one to that resource. The detailed movement routine will be explained in more detail later on. Each *human* knows the current location of all resource-fields and all other humans (Scenario 1). Furthermore, a *human* knows the location of its own *pseudo-agents*. Besides movement the *humans* have additional functions like harvesting, reproduction and harmonization. In scenario 2 *humans* did not belief in *pseudo-agents* but they are able to build a friend list or to attack each other. These functions will be explained in chapter 4.3.

Resource fields

Resource fields are special fields represented by *resource-agents*. Agents are used due to programming and evaluation reasons.[24] Initially these agents are randomly distributed in the environment and cannot move. All fields in the environment are colored white in the first and

[21] An "open world" could be imagined like a world globe. If anything leaves one side of the square it will be displayed on the other side.

[22] In a "closed world" it is not possible to cross the boarders.

[23] In Scenario 1

[24] Agents can interact with each other and are active within the simulation. This method of using agents helped implementing the simulation-process and is useful for the evaluation.

black in the second scenario. Resource fields have different amounts of energy represented by a set of colors (light green for low energy to dark green/black for high energy). They can be harvested[25] by *humans* (that increase their own energy level) and their energy-level increases with a given probability each tick up to a maximum value (their color is adjusted to the energy-level).

Pseudo Agents

> *"… each agent independently (mis)believes in a small set of randomly generated pseudo-agents."* (Doran 1998, 5).

Each *human* believes in a set of 10 *pseudo-agents*[26] within the current initial simulation. This value is also implemented by a variable changeable by the observer. Within the implementation *pseudo-agents* do not move automatically. Only during the harmonization process *pseudo-agents* can jump to another coordinate within the environment.

Fatal zone

The fatal-zone is an area where the *humans* immediately die on entrance. The fatal-zone is represented by one agent located in the middle of the environment. According to Doran the zone should have a radius of a quarter of the environmental width. The diameter of the fatal zone is implemented as a variable factor and can be changed by the observer. The visualization of this zone is realized by a red transparent circle. The location of an agent in the fatal zone is checked by the method "check-death" in the current implementation.

4.3 Model-rule-set and used functions

Like mentioned before it is important for a simulation to translate the given information (behavior of agents, information about the environment, all model related assumptions, etc.) from the natural language, into an explicit and formal language. Doran explicitly describes the rules for scenario 1:

- Initially, n agents and m resources are randomly distributed in the environment.
- Resources have fixed locations in contrast to agents.
- Each agent has an energy level that declines over time.
- Harvesting a resource can restore the energy.
- Resources renew periodically to a maximum value.
- Agents can die if their energy-level declines to zero or if they enter the "fatal" zone.

[25] The energy-level of the field declines to zero.

[26] The value can be changed for each simulation with the help of a slider on the Simulation Interface.

- In each time unit the main actions are to harvest a resource-field (if the agent is located on a resource field), a small random move and a possible target movement towards the next field. This movement is restricted by the rule that the agent must be the closest agent to the resource.
- Agents are aware of the current location of all other resources and agents in the environment.
- Humans do misbelieve in the existence of their own set of other agents that do not really exist[27].
- The *pseudo-agents* cannot move or consume resources.
- Any *human* can reproduce with a small probability in each time unit.
- The set of pseudo-agents is handed on from parents to children with random variation according to the parents' energy level.
- *Humans* can harmonize their pseudo-agents by taking over the belief of another agent in the nearest neighborhood (first way of harmonization).

In the current implementation further rules are implemented:

- The energy level of an agent is between 0 and 100.
- Patch color is white in general with different shades of green for resources.
- While reproducing parent and child share their energy.
- If an agent dies it will be removed from the environment as well as all its related *pseudo-agents*.
- An agent loses one energy unit during a random move.
- An agent loses energy units in the amount of the distance (rounded to the nearest integer) to the resource during a target move.
- The target movement towards a resource will only occur if the current energy-level is higher than the resource-distance (rounded to the nearest integer).
- Resource fields have a maximum energy level of 10 and the color is related to the green color scheme of NetLogo. The darker a field is the higher is the current energy level.
- A second harmonization-method is implemented as well, where the compared *pseudo-agents* meet at their middle position is implemented as well.

For the second scenario Doran identifies the kill-process and the friendship-agreement as the main difference to scenario 1. In scenario 2 it is not necessary to use *pseudo-agents*. Another

[27] In the simulation they exist as pseudo-agents but without any function. Secondly they only influence their owner

important fact is that agents only have a limited view of the environment. For the second scenario Doran further defines:

- The probability that one agent adds another agent to its friend list is determined by the observer.
- An agent passes resource information to friends within its view.
- An agent never attempts to kill another agent from its friend list.
- An agent never attempts to kill another agent with a common friend.
- Friendship is not necessarily symmetric.

In addition some rules are especially defined for the kill process:

- The kill process is a sub-process of the target move.
- The energy level determines the outcome of the battle.
- The winner-agent increases its energy level by the amount of the opponent's energy level.
- The winner-agent stays in the environment whereas the looser-agent is removed from the environment.
- Parents never attempt to kill one of its childes and children never attempt to kill its parents.

Besides these rules the following methods are implemented into the current NetLogo implementation.

regrow-grass

Resource fields can be configured in many ways. They can have a fix energy level, the energy level can increase by each tick, the energy level can increase with a probability by each tick, the energy level could increase to a maximum value or is not limited and the energy level could be implemented by other options. Doran describes this process as following. "Resources renew periodically, which implies that there is a maximum carrying capacity for the environment" (Doran 1998, 5). The information out of this sentence was further formalized for the current implementation process as:

- Resources renew[28]:
  - $e_R(t+1) = e_R(t) + 1$
- Resources renew periodically (with a probability in each tick)
  - $e_R(t+1) = e_R(t) + \begin{cases} 1 \text{ with probability p} \\ 0 \text{ with probability } 1-p \end{cases}$

[28] R represent the Resource-Field, e represents the energy-level, t represents a tick and p the probability

- …there is a maximum carrying capacity…
  - $e_R(t+1) = \max(10, \begin{cases} e_R(t) \leftarrow r > p, r \approx U(0,1) \\ e_R(t) + 1 \leftarrow r \geq p, r \approx U(0,1) \end{cases})$

Within this process the global variable "presource"[29] will be updated because there is a probability that a former resource-field with an energy level of 0 (means that this field is not member of "presource") now has an energy level bigger than 0 and is therefore available for harvesting.

random-move

During the simulation each human is forced to do one random movement according to the rules of the simulation. Within the current implementation it is assumed that a human turns around by a random value out of 360° and moves exactly one field forward. The human will decrease its energy level by one unit. There is the possibility that the human will not survive this movement. Therefore this method calls the check-death method that controls if the human has to be removed from the environment.

target-move1 and target-move2

Doran describes this process as an action that can be taken by an agent as follows:

> *"…movement towards the nearest resource to which it believes itself the nearest agent. This movement will only occur if there exists such a resource. The underlying idea is that there is no point in an agent trying to harvest a resource when some other agent is better placed to harvest it."*

Fehler! Verweisquelle konnte nicht gefunden werden. visualizes this process.

In this process the "Active Human"[30] has to find its nearest possible resource. That is not only a single action, but it is the abstract process of subordinated activities. First all resources will be listed and compared to a list with all resources the human will ignore. Condition one (X_1) compares if there is at least one possible resource left. If the condition is false the process terminates directly. In this case the Target move will not occur. Otherwise the closest resource to the active human becomes the "Found Target". Doran describes further that a movement will only occur if no other agent is better located to the resource than the active human. This action will be executed by the "Found Target". The found target lists all humans and *pseudo-agents* (related to the active human) that have the same or even shorter distance to it. Condition X_3 represents the will to live of a *human*. If the movement to the resource will use all the energy of the active *human* the movement will not occur and the process terminates directly.

[29] "presource" equals a list of all resource-fields with an energy level greater than zero

[30] Active Human is the human agent who currently uses the method.

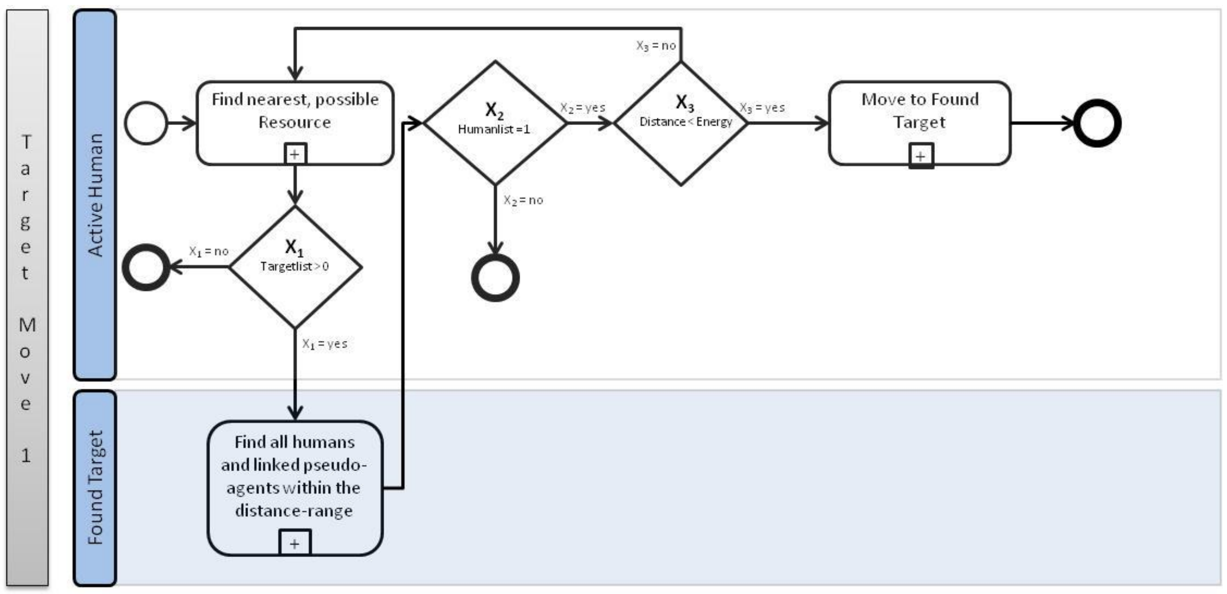


Figure 4-2: The Target-Move-Process for the first scenario according to Doran (1998)

In this case the Target move will not occur. If the energy level of the active *human* is sufficient the movement will only occur if the active *human* is the closest located agent[31] to the resource (condition X_3). If this is false the process starts again with the resource search (the actual found target will be ignored for the rest of the process-run).

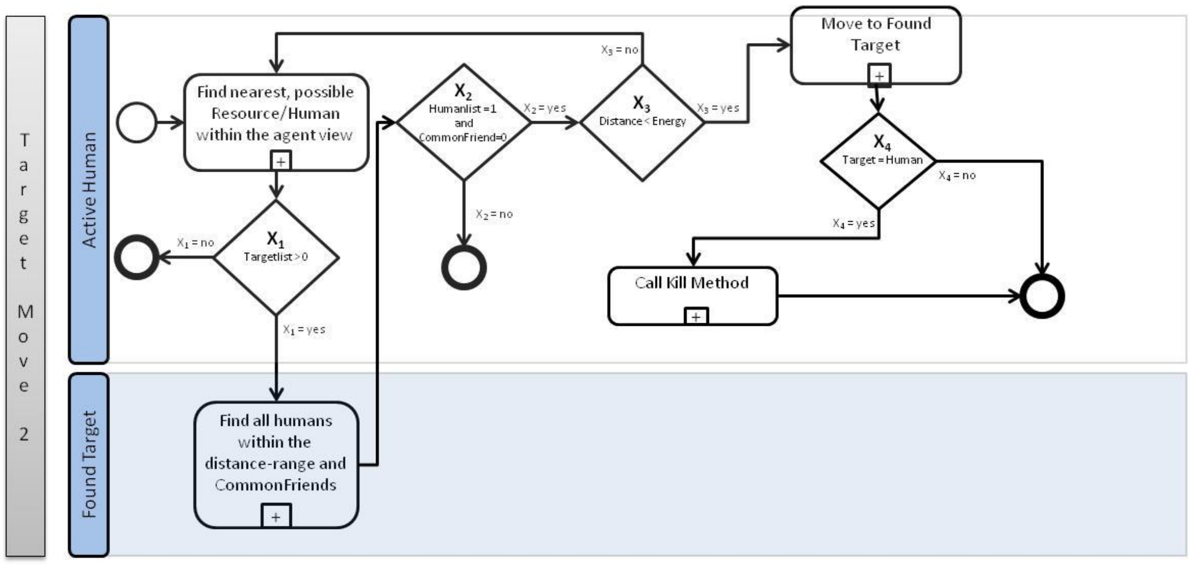


Figure 4-3: The Target Move 2 Process for the second scenario according to Doran (1998)

[31] Agents could be other human-agents or other pseudo-agents related to the active human.

Within the second scenario Doran defines that *humans* can kill each other. That means that *humans* could also be possible targets. Furthermore Doran defines that the kill process will only occur if the attacked human is not a member of the *human's* friend list or has a third friend in common. The Process is visualized within Figure 4-3.

The main process is the same as in scenario 1 but differs in the sub processes and the rules. The target-move process starts with finding the nearest possible resource within the agent-view. Because *humans* can be attacked they are also possible targets. On the other side resource-agents can be friends of the human. This means that all “human-friends” as well as “resource-friends” cannot be possible targets. Condition one stays the same but on the target side it is only needed to find other humans with the same or even shorter distance then the active human. The survival-condition (X_2) stays the same but condition X_3 is complemented by the fact that the found target and active human may not have a common friend if the movement will occur. After the movement there is an additional condition X_4. Here the differentiation between a *human* and a resource takes place. If the target is a resource the process terminates but if it is another human the kill-method (see below) will be executed.

generate_pseudo_agents

As mentioned above each human believes in a small set of *pseudo-agents*. This method is responsible for the creation of these agents. The number of *pseudo-agents* is determined by the variable “No_Pseudo_Agents” that can be changed by the observer. This method is needed when *humans* are created and when *humans* reproduce themselves within the reproduce-method. All created *pseudo-agents* are randomly located in the environment and create a link to their “owner”. Both the link to a pseudo-agent and the pseudo-agents themselves will be hidden initially but can be displayed by a switch on the simulation-interface by the observer.

harvest

At the beginning of each run, all *humans* harvest the energy field on which they are located. This means that an agent consume all energy from the field and increases its own energy level by the same amount. In a more formal language that means if the energy level (penergy) of a harvested field is not equal 0 the *human* increases its own energy level (energy) by the amount of penergy but only to a maximum amount of 100 (the energy level of a *human* can be between 0 and 100). The penergy of the harvested field must be reduced to 0 and the color of the patch must be reduced to an equal lime-green color (even if the human was not able to consume all the energy from the field). Lime-green because the field is still a resource field and is identified by its color. Then the list presource must be updated because other *humans* should not find this resource within the current simulation run.

check-death

This method is a sub-method called by other methods like random move or target-move 2. It removes the *humans* from the environment if the *human* energy level is zero or below or if the *human* is within the fatal-zone. Within scenario 2 the *human* and its entire related links will be removed from the environment with the “die”-command. All human-related *pseudo-agents* must be removed from the environment first, before *human* can be removed. The order is important because otherwise *pseudo-agents* with no relation would stay and the following evaluation of the results could be influenced negatively.

reproduce

For both scenarios Doran describes the option for each *human* to reproduce

> *“…there is a small probability that any given agent may reproduce, that is, create an exact copy of itself located at an adjacent point…”* (Doran 1998).

Further he describes how the heredity of knowledge should proceed. In scenario 1 the pseudo agents should be handed on to the child agent “with a possibility of a random variation when agents’ energy levels are low” and within scenario 2 the friend-list of the parent is handed on from parent to child agent. The current implementation realizes both methods as following: First the energy level is divided by 2 and one child-agent is created as a human. The child should be at an adjacent point of the parent human. Therefore, the child does a random turn within 360° and moves one step forward. It also gets the same energy level as the parent. In scenario 1 the method now calls the “generate_pseudo_agents” because the child-agents need pseudo-agents. These *pseudo-agents* are randomly distributed. Then two lists are generated where the *pseudo-agents* of the parent (harmonize1) and the ones of the child (harmonize2) are listed. Each *pseudo-agent* of the child list is compared with *pseudo-agents* on the parent list. If a random system value is within the range of the energy level the *pseudo-agent* of the child-agent move to the same location as the *pseudo-agent* of the parent.

add-friends

This function is only related to scenario 2. Here, Doran described a non-symmetric relationship. He explains in detail that it could happen that agents meet. Agent X can decide if it wants to add another agent Y to his friend list. This is not necessarily a bidirectional relationship because Y is not forced to add X to his friend list as well. The friend list is important for the models because some “model-rules” refer to the friend list. *Humans* can add other humans as well as *resource-agents*. Within this method a *human* now creates a list of all humans within its agent-view[32] and with a probability (friendliness[33]) the found *human* can be added to the friend

[32] In scenario 2 each agent has only a limited view to the environment.

list. This process must be repeated for *resource-agents* as well. The implementation works with two lists (human-list and resource-list) because in another method (information_transport) the human communicates his known resources to other *humans*. The knowledge about existing resources is saved within the variable "resource memory". While processing this method all found *resource-agents* are added to the resource memory independently if the *resource-agent* was known before or not. To remove all duplicates the ID of the agent must be written into a separate list and then all identified duplicates are removed from the list. This list is the base of the "new" resource-knowledge. The command "remove-duplicates" works with items[34] but the resource memory-list contains turtle-set information. Therefore the transformation into a separate list is needed. After the remove-process the item based list is re-transformed into a turtle based list.

harmonize_1.1 and harmonize_1.2

Doran describes an option that the belief of two neighborhood agents could be harmonized. Within this process one agent takes over the belief about the location of pseudo-agents of his neighbor if they differ from its own. The current implementation realizes this option in the way that one active *human* seeks for his nearest neighbor. Then the two lists with the pseudo-agents of both *humans* will be compared. A loop compares if the nearest own pseudo-agent has an equal location as the closest pseudo-agent of the neighbor-*human*. If they match nothing happens. If they did not match the own pseudo-agent moves to the location of the neighbor *pseudo-agent*. This happens to all pseudo-agents of the active *human*. Afterwards the two *humans* have the same believe. In addition to this process the implementation has the opportunity of a second type of harmonization. In general this process works the same way. The difference is that harmonization does not mean to take over the belief of the neighbor. Both *pseudo-agents* will meet in their "middle position"[35]. The selection between the two types of harmonization could be done in the observer view of the simulation.

information-transport

Doran describes that an agent passes its information about resource-locations if the opposite agent is within the agent's friend list. This action is realized in the simulation in that way that an agent passes the information about his known resources to all "friend-agents" within his view. The agent takes the resource-list of its friend, adds its resource list and removes all duplicates. The changes will only be made in the partners list because it is only described that an agent passes the information. This information-transport only occurs in scenario 2.

---

[33] Represent the probability which the humans add another agent to his friend list.

[34] The list consists of stings with single information like the „who"-number of an agent.

[35] The middle of the two former positions of the pseudo-agents.

kill

Doran describes that a *human* can kill another *human* under different circumstances. But there is no closer information about the kill process itself in the description. The process is restricted by some model rules that describe circumstances when a kill-action is not allowed. The process is therefore implemented as a sub-process of “target-move”. In detail the process compares the energy level of the attacker with the energy level of his opponent. If the attacker’s energy is equal or higher than the opponent energy level the attacker gets all energy from the opponent. The energy-restrictions[36] are valid here as well.

[36] Energy has a maximum value of 100.

5 Analysis

The analysis is divided into several parts. At first the syntax-check is explained. Secondly the functionality of the source code is checked in the interface view. The evaluation of the model itself is performed on basis of several analyses of simulation runs (one run for each variable to find a matching variable-set and 40 runs with the same variable-set to check factors like validation of stable population, impact of *pseudo-agents*, development of cults and the average belief in dead *humans*). The results are transferred into Excel format and an excel-based data-evaluation takes place.

5.1 NetLogo code validation

As explained in the theoretical background chapter NetLogo offers an automatic syntax check. This check is executed each time the user switches from the "procedures-view" to the "interface-view" or by a manual execution. If there are **syntactical errors** (e.g. if used variables are not initialized or if other syntax rules are not applied) the observer[37] gets an "Error message".

Functional errors can be checked in the "Interface-view". Plots like "Humans" display the number of the current *human* population. The equal plot "human" visualizes the same information in a graphical way. If there are differing values or values that violate given constraints (like negative values for the population) it is a sign for a functional error. Consequently the source code must be checked if some semantically wrong interpreted commands are the reasons for the functional error. Loops can also be identified during the execution of the simulation. Some output-commands (debugging points) from the source-code (like show a variable at a given point of the simulation) results that this information is edited in the "Command Center Box" of the interface view and help to debug the source code. During the simulation the observer can interact with single agents, change parameters of the simulation or request information about the status of the simulation or of different variables.

[37] Observer in the case means the person responsible for the source code. This person may also be at the same time the simulation user.

5.2 Evaluation of Scenario 1

The evaluation of both scenarios takes place in four steps:

1. Variable definition
2. Simulation run
3. Transfer results to Microsoft Excel
4. Excel-based evaluation

First, the variable-space for the needed variables must be defined. NetLogo offers an internal tool “BehaviourSpace” where experiments can be launched. Each defined variable must be specified in the experiment as well as a corresponding experiment-replication-number, the needed reporter (here for example the population could be counted after each tick), the execute commands and the time limit of the experiment. If all specifications are filled the experiment can be executed. It will run all possible combinations of variable-sets and exports the results of the watched variables into a “.csv”-file. This file can be opened in Microsoft Excel and the data is processed.

The main target is to find a combination of variables for a stable population[38]. In this evaluation it is assumed that the initial population has a fix value of 100 *humans* and the simulation should run 1,000 ticks. Furthermore each *human* believes in 10 *pseudo-agents* and the diameter of the fatal zone equals 50% of the width of the world. Within the simulation each energy-patch can have a value between 0 and 10. *Humans* lose at least one energy-unit each tick by random movement and a number of units according to the distance to the target. Therefore, the harvest variable is tested with the values of one, two, five and ten. The harvest multiplier is intended to balance the high-energy consumption implicated by movement and the comparable low energy values of the resource fields. In nature, most resources are harvested once a year. For the grow-factor of the energy-fields this means that with a probability of 1/4 (each 4th quarter of a year) the energy should grow by one unit. In the evaluation a renewing-energy-factor is tested to find an adequate combination of variables (12%, 25% and 50%). The maximum number of possible energy-patches throughout the environment is 1098. The evaluation is made with numbers of 140, 270 and 540 energy-fields. Even in the real world not every square is used economically. It is allowed that *humans* reproduce themselves. The simulation is executed with reproduction-probabilities of 0.01%, 0.1% and 0.8%. For the evaluation of scenario-one all relevant functions, rules, elements and explained agent-behavior will be executed.

[38] A stable population is a population that finds a level of a stable size around 30 to 60% above the initial population size.

The combination of all variables resulted in a number of 108 runs of the simulation. During the simulation, the number of active *humans* has been logged. The results of these runs give an indication of the survivability of the population with appropriate combination of variables. Five groups can be identified:

Group 1: No humans survived

No *human* survived until tick 1,000

Group 2: Reduced Population

After tick 1,000 the population has a semi-stable population between 0 and 100 *humans*.

Group 3: Population has stabilized

The population has stabilized at a level above 60% over the original population. Population-size is between 101 and 160 *humans* after tick 1,000.

Group 4: Population has increased stabilization

The population has stabilized at a level of 161 to 260 *humans* at tick 1,000.

Group 5: Population rises sharply

The population has risen sharply to a stable level above 260 *humans* or to a stable rising curve.

From the frequency of the grouping it is shown that:

- 73% of the variable combinations belong to Group 1,
- 11% of the variable combinations belong to Group 2,
- 6% of the variable combinations belong to Group 3,
- 2% of the variable combinations belong to Group 4,
- 8% of the variable combinations belong to Group 5.

Variable-Combinations of Group 3

[run number]	45	51	69	75	87	102
harvest-factor	2	2	5	5	10	10
renewing-energy	25	50	25	50	12	50
initial-energypatches	540	270	270	140	270	140
reproduce-factor	0.8	0.8	0.8	0.8	0.8	0.8
initial-population	100	100	100	100	100	100
No_Pseudo_Agents	10	10	10	10	10	10
size-of-fatal-zone	50	50	50	50	50	50
[reporter]	count humans	count humans	count humans	count humans	count humans	count humans
[final]	128	124	160	139	122	149
[min]	78	80	75	74	69	68
[max]	161	128	184	147	151	186
[mean]	131	104	136	116	114	134
[steps]	1000	1000	1000	1000	1000	1000

Table 5-1: Variable-Combinations out of Group 3 (own evaluation-table)

For further evaluation, group three was selected. In this group there is a stable population which is increased with a maximum 60% above the initial population. These possible variable-combinations are listed in Table 5-1.

Only the reproduction-probability of 0.8% leads to a favored stable population. This factor seems to match the death rate of the *humans*. The comparison of run number 69 and run number 75 shows that the number of energy-patches (270 and 140) can be balanced with a change of the renewing-energy-factor (25 and 50). Furthermore with an extreme harvest factor of 10 there is no need for a high number of energy-patches if the renewing energy-factor is high. Combinations consisting of extreme values like 50% of the renewing-factor together with a high harvest-factor have been sorted out. Finally, for later evaluation the run-number 45 is selected with the following variable-combination:

- Run number: 45
- Renewing-energy: 25
- Initial-energy patches: 540
- Reproduce-factor: 0.8

This combination of variables was executed 50 times to make sure that the results were stable. Figure 5-1visualizes the mean population after 1,000 ticks out of 50 runs with the same variable-combination.

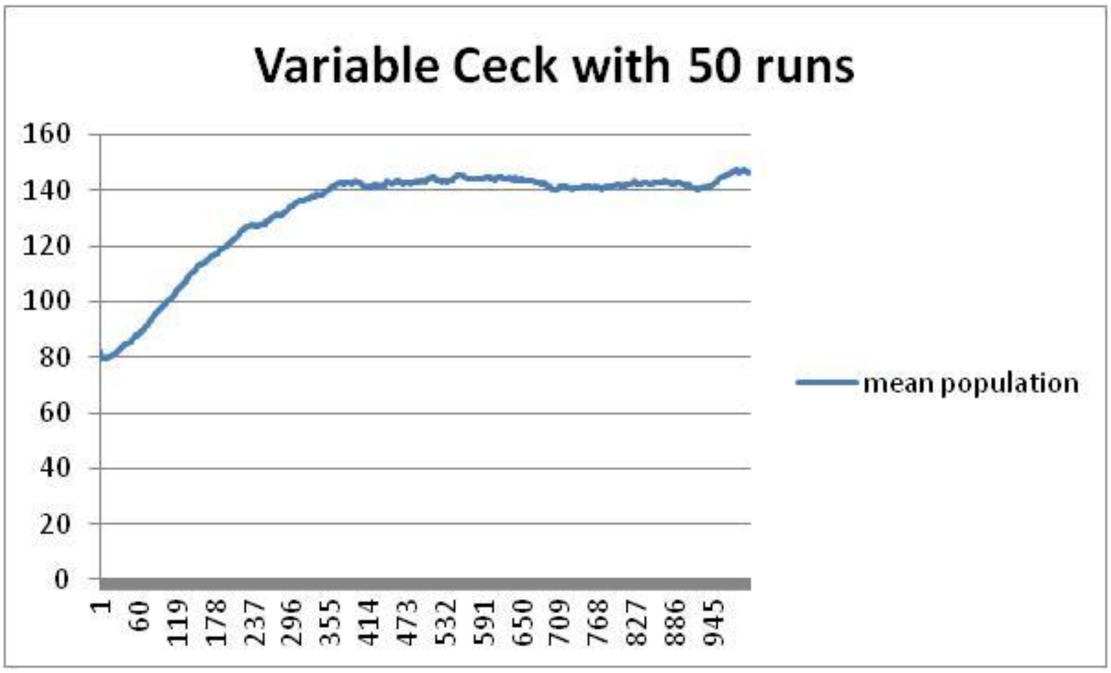


Figure 5-1: Variable Check with 50 runs (own evaluation-figure)

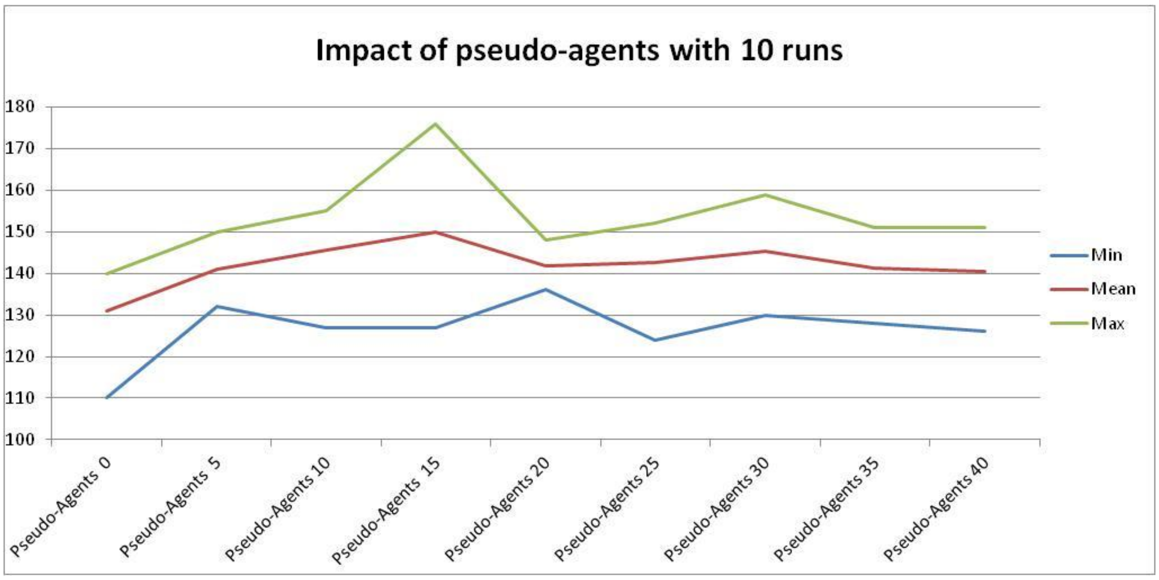


Figure 5-2: Impact of pseudo-agents with 10 runs (own evaluation-figure)

The title of the model is “Simulating Collective Misbelief”. Therefore, the further evaluation takes a closer look at the impact of *pseudo-agents*. So far each *human* had a fix number of 10 pseudo-agents. In the next evaluation step this variable is changed. This evaluation is made twice. First, the variable gets the value 0, 5, 10, 15, 20, 30 and each value is executed 10 times with 1,000 ticks.

Figure 5-2 shows that there might be a relation between the number of *pseudo-agents* and the size of the survivor population[39]. Until the number of *pseudo-agents* reaches a value of 15 the survivor population-size increases. Together with the min and max values of the 10 runs per variable it is shown that the difference between the population-sizes is not as high as Doran explained.

> *The effect is to maintain a substantially (about 50%) greater agent population than would otherwise be the case (as determined in control trials).* (Doran 1998)

To verify the results this experiment has a control trial with 0, 5, 10, 20, 25, 30, 35, 40 *pseudo-agents* and 40 times runs each variable with 1,000 ticks.

[39] Survivor population means the number of living agents at the end of the simulation.

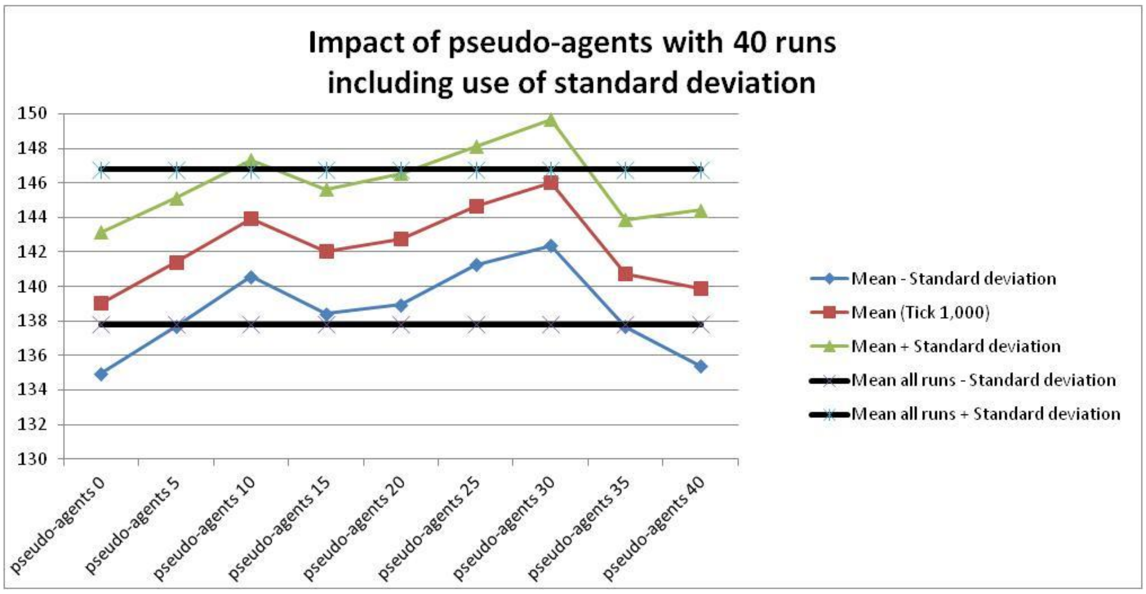


Figure 5-3: Impact of pseudo-agents with 40 runs and standard deviation (own evaluation-figure)

In Figure 5-3 the min and max values are replaced with the values of mean plus and mean minus 1.96 standard deviations for the individual parameter combinations and the respective overall values. Nearly all values are within this 95% confidence corridor which leads to the impression that the number of *pseudo-agents* (in contrast to Doran’s results) has no statistical relevant impact to the survivor population-size of the *humans* in the Simulation. The initially observed dependency between the number of *pseudo-agents* and the survivor population size seems to be random.

According to these results of the current evaluation with the specific used variables and the current NetLogo implementation, the hypothesis postulated by Doran can be disproved in favor of the null hypothesis (the number of *pseudo-agents* / or the (mis-) belief of the *humans* have no impact to the survivor population size).

Doran mentioned the possibility, that a specific set of (mis-) belief can be spread in the population.

“… beliefs are typically handed on from 'parent' to 'child', with a possibility of random variation when agents' energy levels are low.” (Doran 1998)

This implies that with the reproduction of a *human* the belief will partly be handed to the child and together with the missing part (the rest to complete the 10 *pseudo-agents* per *human*) a new belief (new set of pseudo agents) is created. In the current implementation each *human* has his own *pseudo-agents* but it is possible that more than one *pseudo-agent* is located at one patch (more than one human believe in a *pseudo-agent* at this patch). For the evaluation all patches with more than one *pseudo-agent* will be counted and grouped in relation to the overall population. Further groups will be formed based on the following criteria:

Cult 1	The number of pseudo-agents is < 10% of the population
Cult 2	The number of pseudo-agents is >= 10% and < 20% of the population
Cult 3	The number of pseudo-agents is >= 20% and < 30% of the population
Cult 4	The number of pseudo-agents is >= 30% and < 40% of the population
Cult 5	The number of pseudo-agents is >= 40% and < 50% of the population
Cult 6	The number of pseudo-agents is >= 50% and < 75% of the population
Cult 7	The number of pseudo-agents is > 75% of the population.

Doran's statement that a set of *pseudo-agents* can be spread through the population can be reproduced with the current implementation. After a replication of 50 runs with the same variable set it is shown that (after 1,000 ticks) the belief in the location of a *pseudo-agent* can be spread through the population. Figure 5-4 shows that after 1,000 ticks, there are about 10 patches, each of them occupied by 20-30% of the suspected pseudo-agents.

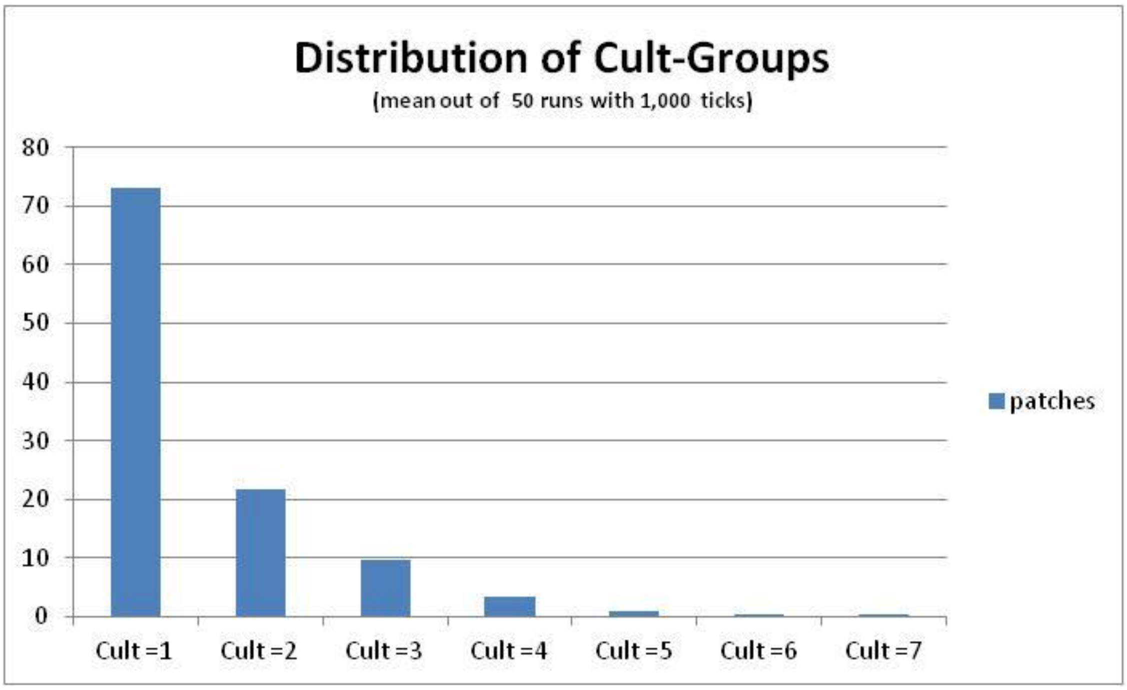


Figure 5-4: Distribution of Cult-Groups A (own evaluation-figure)

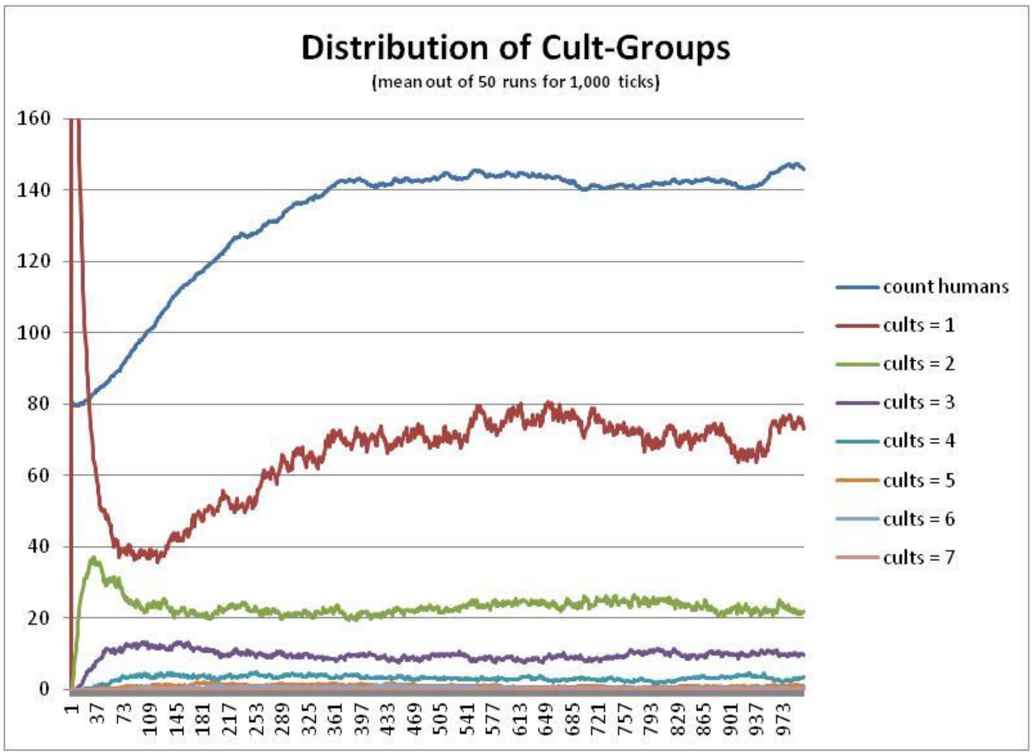


Figure 5-5: Distribution of Cult-Groups B (own evaluation-figure)

The number of patches in “Cult = 1” can be explained by the fact that initially the *pseudo-agents* of each *human* are randomly distributed in the environment. Therefore at the beginning of a simulation there is a low probability that more than 10% of the population share the belief in the existence of a pseudo-agent at the same location. The harmonize process (where one *human* takes over the belief of another *human*) and the reproduce process (where the *parent human* hands over his belief to the *child-human*) are responsible for sharing and spreading the belief throughout the population. These two effects lead to a falling curve of “cults = 1” and an increasing curve of “cults = 2”. Around tick 145 the increasing population leads to an increasing curve of “cults = 1”. New *humans* only partially take over the pseudo-agents of the parent-agent. This causes an increasing number of fields belonging to “cults = 1”. This effect is also evident in the development of the various cults over the 1,000 ticks as visualized in Figure 5-5.

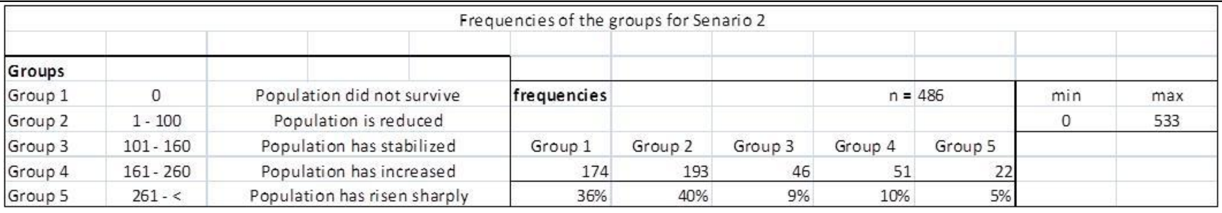
5.3 Evaluation of Scenario 2

Scenario 2 focuses on the so-called “category error”. *Humans* can add *resource-agents* to their friend list and treat them like a *human agent*. Consequently, this leads to an abnormal behavior of the *humans*. The *human’s* friend list stands for the (mis-) belief of the *humans*.

Like in scenario 1 the evaluation in scenario 2 follows the same schema.

The first step was to find a suitable combination of variables for a stable population[40]. Additional to the changing variables in scenario 1 (renewing energy, number of energy fields, harvest factor and reproduce factor) the friendliness[41] and agent view[42] factor is taken into account as well. Possible values for friendliness are 4, 8 and 12 (4%, 8% and 12% probability to add another agent to the friend list) and for agent view 5 and 10. The evaluation is realized by executing the simulation for all possible combinations (486 runs). The main focus is to find a combination where the population has a stable level 30% to 50% above their initial population size.

Based on the results, the same five groups as in scenario 1 can be identified (0, 1 – 100, 101 – 160, 161 – 260, 261 - < population size). The frequencies are visualized in Table 5-2. As in scenario 1 group 3 has a frequency below 10% but the other group frequencies differ.



Frequencies of the groups for Senario 2

Groups									
Group 1	0	Population did not survive	**frequencies**			n =	486	min	max
Group 2	1 - 100	Population is reduced						0	533
Group 3	101 - 160	Population has stabilized	Group 1	Group 2	Group 3	Group 4	Group 5		
Group 4	161 - 260	Population has increased	174	193	46	51	22		
Group 5	261 - <	Population has risen sharply	36%	40%	9%	10%	5%		

Table 5-2: Frequencies of the groups for scenario 2 (own evaluation-table)

Group 3 consists of 46 different combinations of variables and is reduced further. Run number 10 and 11 differ only in the agent view but this difference leads to a greater population size of about 20 *humans* (if the *human* has a lager view). In general, a low harvest factor only leads to a sufficient population size if other factors like renewing energy or reproduce factor are at their maximum. This evaluation is the result of just one execution for each variable combination.

Finally, the following variable-combination is selected for the validation of the population size and other factors:

- Run number: 87
- Renewing-energy: 25
- Harvest factor: 2
- Initial-energy patches: 540
- Reproduce factor: 0.8
- Friendliness: 8 and 0
- Agent view: 5

[40] Like in scenario 1, a stable population is reached if the population size has a stable level of around 30 to 60% above the initial population.

[41] Friendliness is a variable for the probability that a human adds another human or resource agent to its friend list.

[42] Agent view equals the number of patches a human is able to “see” in each direction.

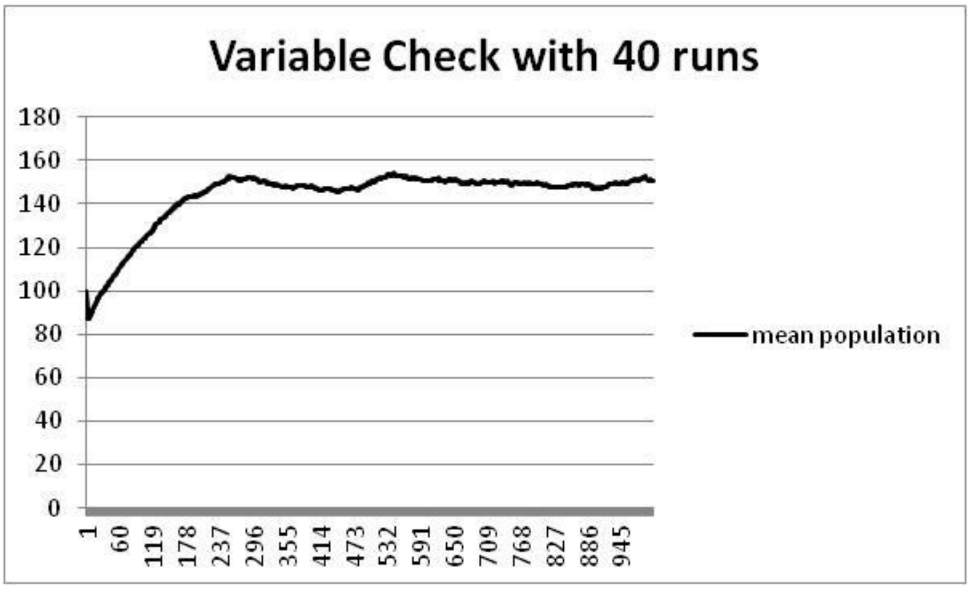


Figure 5-6: Validation of the population size (own evaluation-figure)

This combination of variables was executed 40 times to ensure the results were stable.

At the beginning the population size is reduced to around 85 *humans* within the first 30 ticks (cf. Figure 5-6). This can be explained by the fact that at the beginning of a simulation the number of friends is low. Consequently the number of attacks is high because *humans* take other *humans* as possible targets. Figure 5-7 visualizes the development of attacks and shows that per period 0.1 attacks occur (except at the beginning of the simulation).

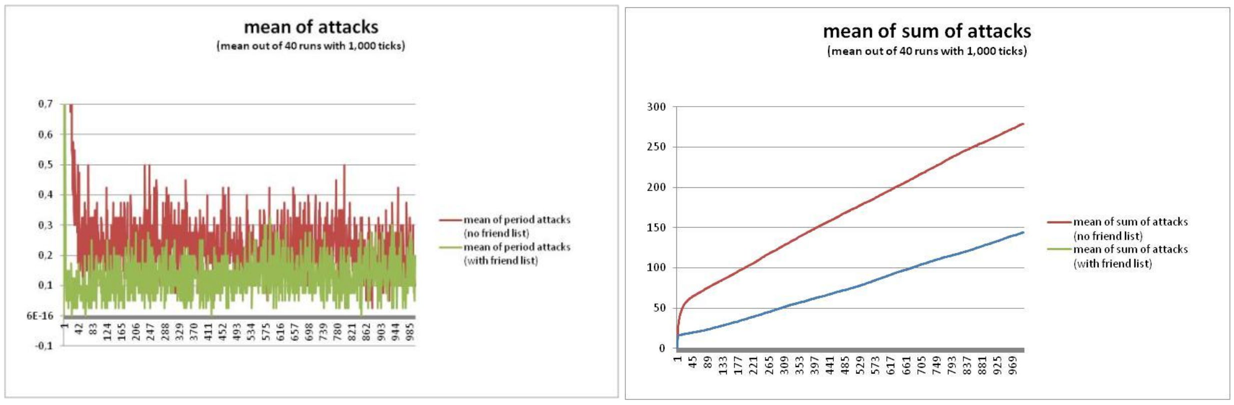


Figure 5-7: Development of attacks (own evaluation-figure)

Furthermore, in Figure 5-7 the difference between the collective (mis-) belief and the behavior without using the friend list (the usage of the friend list illustrates the (mis-) belief) is pointed. Here it is shown that the attacks per period are less if a friend list is used. This impression gets more strengthened if the diagram of the added attacks is taken into consideration as well. This positive effect out of the (mis-) belief results also in bigger population size.

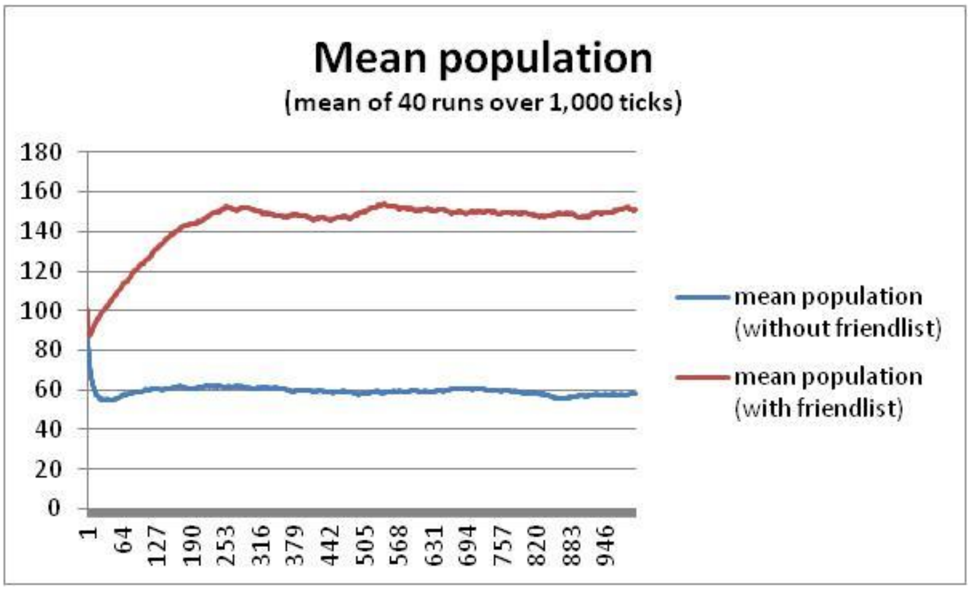


Figure 5-8: Development of attacks (own evaluation-figure)

Figure 5-8 displays that the population reaches a stable size at around 150% (initial population size was 100). Without a friend list the population also reaches a stable size but at around 40% below the initial population size. In this case Doran's general thesis that (mis-) belief has a positive impact on the population could be reproduced with this scenario.

Doran also evaluates the phenomenon of the cult development. As in scenario 1 the combination of "cult-heads" stays the same (0%, 1-10%, 11-20%, 21-30%, 31-40%, 41-50%, 51-75%, 75%< *humans* belief in the friendship). One main difference is the fact that *humans* as well as resource-agents can become a cult head. Figure 5-9 shows that there are similarities between the development of the "Cult-Heads" in scenario 2 and the development of "Cult-Groups" in scenario 1. The reasons for this development are also nearly similar. At the beginning of the simulation many single *humans* are distributed in the environment with an empty friend list. The friend lists grow larger with every tick and the corresponding reproduction cycle such that at first many small cults raise and later bigger cults came up.

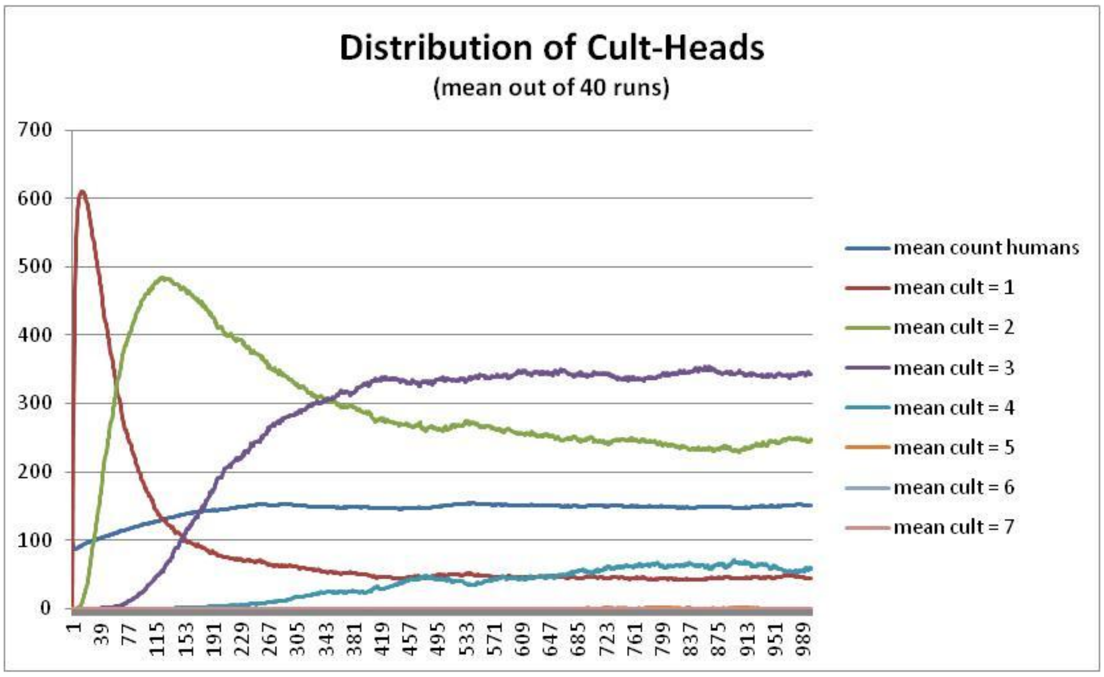


Figure 5-9: Distribution of Cult-Heads (own evaluation-figure)

If a *human* dies the cult of this agent is removed from the evaluation. But with a NetLogo based additional evaluation it can be shown that it is possible to have a cult among dead *humans*:

At tick 0 the simulation is asked for some variable-states:

```
observer[43]> ask human 586 [show ticks show friendlist show cult-head-status show count humans]
        (human 586)[44]: 0
        (human 586): [586]
        (human 586): 1
        (human 586): 100
observer> let cu 0 ask humans [if member? 586 friendlist [set cu cu + 1]] ask human 586 [show cu]
        (human 586): 1
observer> ask human 586 [ let compare friendlist let survivorpopulation [who] of humans  let resources
[who] of resource-agents foreach resources [set compare remove ? compare] foreach survivorpopulation
[set compare remove ? compare] show compare]
        (human 586): []
```

At tick 14 the simulation is asked for some variable-states:

```
observer> ask human 586 [show ticks show friendlist show cult-head-status show count humans]
        (human 586): 14
        (human 586): [35 399 182 208 196 104 351 447 67 78 352 457 163 228 116 153 451 357 600
        128 261 19 556 327 404 51 148 199 586]
        (human 586): 1
        (human 586): 96
observer> let cu 0 ask humans [if member? 586 friendlist [set cu cu + 1]] ask human 586 [show cu]
        (human 586): 5
observer> ask human 586 [ let compare friendlist let survivorpopulation [who] of humans  let resources
[who] of resource-agents foreach resources [set compare remove ? compare] foreach survivorpopulation
[set compare remove ? compare] show compare]
        (human 586): []
```

[43] Observer is the command from the observer to the simulation

[44] () is the answer from the answering agent. In this case human 586

At tick 35 the simulation is asked for some variable-states:

```
observer> ask human 586 [show ticks show friendlist show cult-head-status show count humans]
        (human 586): 35
        (human 586): [586 411 413 362 38 597 426 517 139 383 64 236 523 37 57 35 399 182 208 196
        104 351 447 67 78 352 457 163 228 116 153 451 357 600 128 261 19 556 327 404 51 148 199
        586]
        (human 586): 3
        (human 586): 101
observer> let cu 0 ask humans [if member? 586 friendlist [set cu cu + 1]] ask human 586 [show cu]
        (human 586): 24
observer> ask human 586 [ let compare friendlist let survivorpopulation [who] of humans  let resources
[who] of resource-agents foreach resources [set compare remove ? compare] foreach survivorpopulation
[set compare remove ? compare] show compare]
        (human 586): [597 600]
```

The above displayed command-line abstract from the NetLogo simulation has the same structure in all three code excerpts. First, the observed *human* is asked to show the current tick, its current friend list, its cult-head-status and the number of current *humans* alive in the whole environment. The second command asks how many other *humans* believe that the observed human is a friend. With the last command the observed *human* shows the *humans* from its own friend list, which are not alive.

> The first block shows that the *human* starts with an empty friend list (except of the *human* itself[45])
>
> At tick 14 the *human* has added some friends. The agents 1 – 540 are resource agents based on the simulation setup and higher agents are other *humans*. 5 other humans treat the observed human as a friend (resource agents are not able to setup a friend list).
>
> The last block indicates that it is possible that a cult is based on a dead *cult-head human*. The observed *human* 586 believes in the cults of *human 597* and *human 600*. Both died some time before tick 35.

The potential of dead *humans* to become a cult-head is quite low because other *humans* cannot find them and add them to their friend lists. But the friend lists are not deleted so there is a possible but unlikely scenario where a dead *human* becomes a cult-head after death. Only if other cult heads die and the members of the dead cult head agent stay alive the cult among the cult head will alive.

After 1,000 ticks each *human* believes in 36%[46] dead *humans*. Figure 5-10 displays the development of the percentage of dead *humans* in the friend lists of *humans* in the environment. In the beginning there is a strong raise of the curve as friendships are not that wide spread and many single cults exist. After tick 500 the curve seems to enter a semi stable level. This might

[45] The human must be at his own friend list otherwise he will find himself as a possible target.

[46] Among all humans in the friend list, 36% of them are not alive.

be due to the fact that bigger cults are introduced and with many *human* friends a potential resource (attackable *humans* as a potential resource) is no longer available.

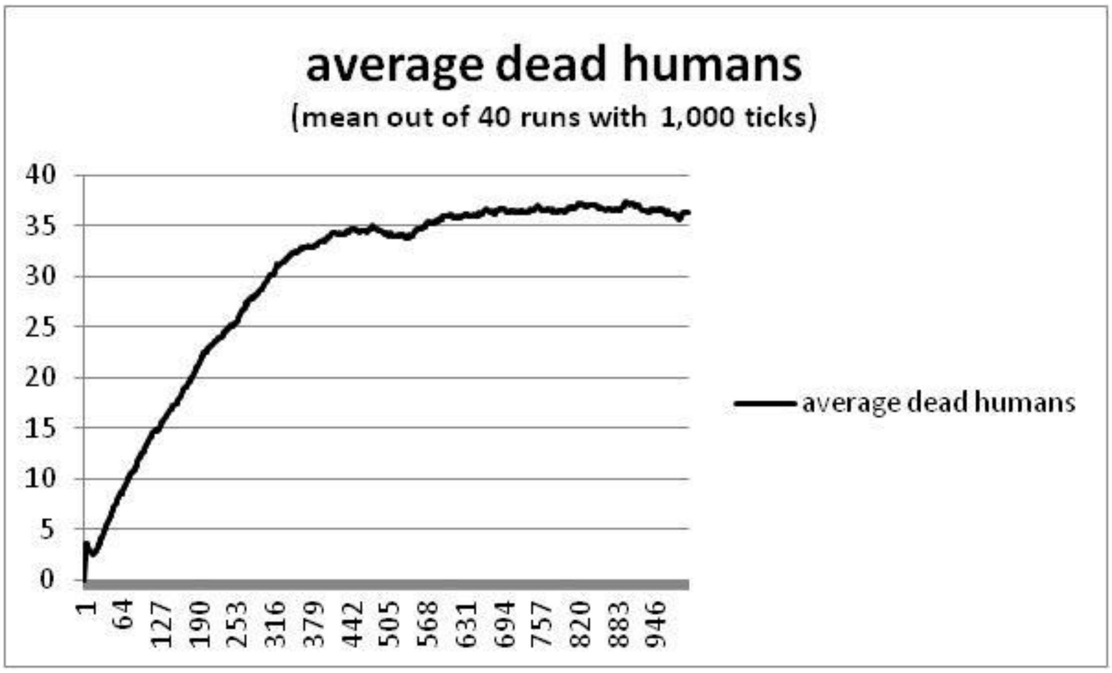


Figure 5-10: Average dead *humans* in the friend list (own evaluation-figure)

The above described evaluation of scenario 2 is based on the approach that a *human* starts in the environment with an empty friend list. Even after reproduction the *child human* gets an empty friend list (except of itself and its *parent human* as the only friends). This approach contains the idea that not every friend of the parent is necessarily a friend of the child.

Another approach is that the child *human* receives the complete friend list from the parent *human*. This approach reflects the idea that a child is born into a cult and starting with that basic cult and it adds new friends. Figure 5-11 displays the difference between these two approaches based on the example of the surviving population.

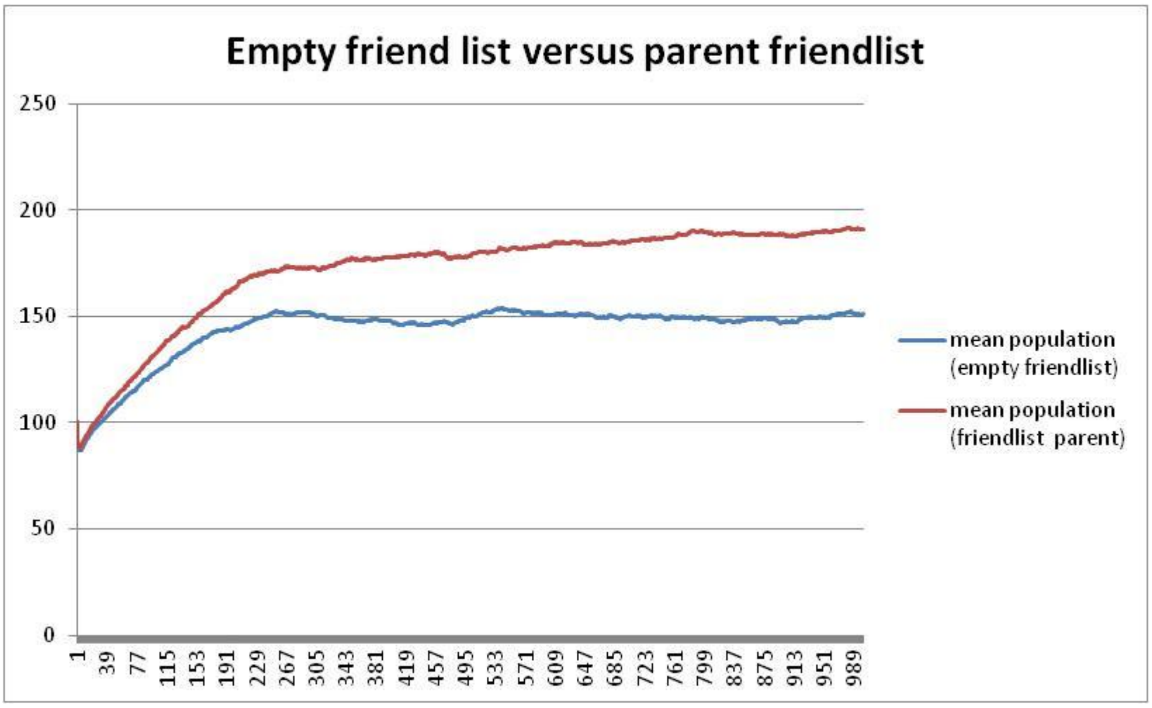


Figure 5-11: Empty friend list versus parent friend list (own evaluation-figure)

6 Conclusion

After the analysis of the NetLogo simulation this chapter references back to the results of Doran and present a critical appraisal of the findings in this work. Furthermore this chapter emphasizes some NetLogo-specials during the implementation. Finally this thesis hints towards several ideas on how the model could be changed or extended for further work.

6.1 Linking of this thesis to the results of Doran

The evaluation in chapter 5 showed that some results of Doran can be reproduced by using the implemented NetLogo model. In this section the results of Doran will be linked to the evaluation results of this thesis.

In the first scenario Doran mentioned that the model enters a semi stable state

> *This semi stable state was the basis of the evaluation of this thesis.*

with a demonstrably enhanced agent survivability caused by collective misbelief in a small set of *pseudo-agents.*

> *The evaluation shows that with the chosen combination of variables the number of pseudo-agents does not have any significant impact to the population survivability. Like mentioned in the evaluation chapter the population size did not differ that much by using different amounts of pseudo-agents. Nearly all population sizes were within the 95% confidence corridor of the standard deviation.*

Doran quantifies the positive effect by 50%.

> *This big positive effect could not be reproduced. Therefore the existence error has no significant impact to the agent survivability.*

Additional to the results of Doran this thesis also evaluates the development of cults with the conclusion that there is a trend to collective (mis-) belief in the environment and that a strong increasing population size assists a rising curve of small cults.

Doran mentions that the model tends to favor the building of small cults.

> *Depending on the classification of cults Doran might be right. The results of this thesis show in general that at the beginning of the simulation small cults have a high frequency. Over time their number declines and bigger cults come up.*

Furthermore Doran presents a concrete output of his simulation. He raises the idea that *resource-agents* incorrectly can be convinced as a friendly agent (attribute error), the idea of cult building and the idea that it is possible to have a cult among a death agent. But he also mentions that a *resource-agent* might be more efficient as a cult-head than a death agent.

These results were reproduced with the current implementation.

In addition to Doran's evaluation this evaluation a closer linkage to the evaluation results of scenario 1 and focuses more on the impact of the attribute error.

Finally, Doran mentions that different combinations of variables and different settings of parameters may lead to very different outcomes.

By selecting the possible combination of variables for scenario 1 and scenario 2 the main focus was to find a combination which results in a stable population size around 150% of the initial population. Here it is shown that several combinations fit into this requirement and that there is not "the only" one solution. Otherwise it is shown that another set of variables results in a permanent increasing population or permanent decreasing population.

As the main conclusion of this work it is found that some conclusions presented by Doran could be reproduced and some not. This might be caused by a different set of variables, a different behavior rule set of the agents or the different simulation platform.

6.2 NetLogo challenges

Some functions in NetLogo could not be implemented completely. Doran describes that target-movement for example only occurs if the agent was the closest agent to the resource (Doran 1998). This requirement was coded in the way that the potential resource generates a list consisting of all *humans* with the same or even shorter distance to itself. A second list encompasses all *pseudo-agents.* Here not all *pseudo-agents* listed. Only the *pseudo-agents* of the active agent are of interest. The condition is that the "human list" must consist of exactly one *human* (the *active human*) and the list with *pseudo-agents* must be empty. Then the active *human* can be sure that he is the closest agent to the *human*. Doran's requirement is therefore not implemented as a single action processed by one agent. It is a communication process where the *human* asks the resource for some information. Based on the reply it decides if the movement takes place or not.

Another NetLogo-functionality is the opportunity to work with lists where each element itself is a list. In the implementation the list consist of different agent-sets. The "information_transport" function for example writes a list with known resource-agents from one agent into the list of another agent. Now it can happen that the new list has duplicates. NetLogo provides a command to remove duplicates but this command does not work on lists with elements consisting of lists. Therefore the new list is changed into an item-based list (the list only consists of the "who"-numbers of the resource-agents). After that the "remove-duplicates"-command is applied. Then the list will be "re"-changed into the agent-set based list.

The model of Doran is enriched in the current implementation by the component of providing suicide. *Humans* move directly to the resource instead of doing just one step towards the resource. The *human* loses energy corresponding to the amount of the round distance to the resource. Without the limitation that suicide is forbidden it can happen that the agent loses more energy by moving to the resource than its own energy level allows for. An if-condition in the source-code prevents this behavior. It checks if the current energy level is high enough to reach the resource.

6.3 Outlook

Like mentioned above the targets of this thesis were fulfilled. Still, bases on a closer look on the scenarios and the described simulation process several questions came up.

The process of harmonization of the *pseudo-agents* in scenario 1 is described as a taking over process. *Human* one takes over the believed in *pseudo-agents* of human two. Theoretically, it is possible that a set of *pseudo-agents* will be spread throughout the population. Doran mentions this effect in his paper but he does not describe the corresponding evaluation. This effect is not necessarily the case. If in the same round the second *human* will also take over the believed *pseudo-agents* of another agent (which is not necessarily a *human* one) his originally believed set of *pseudo-agents* is not increased. Based on the idea that only the strongest will survive, the harmonization process could be linked to the energy level of the *humans*. Within the harmonization process the “stronger” *human* would give its believed set of *pseudo-agents* to the “weaker” *human*. Another idea could be that the *humans* initially do not receive an individual set of *pseudo-agents*. The number of sets of *pseudo-agents* in this scenario is limited and each *human* initially belongs to one set randomly. In scenario one the spreading of each set of *pseudo-agents* could be followed up. This idea could also be used in scenario two with a rule-extension that *humans* with the same set of *pseudo-agents* cannot attack each other and information will only be shared within the same “believe-group”.

Doran did not mention the phenomena that a *human* may lie for its own benefits. Scenario 1 will not be subject to this idea because all *humans* are aware of the location of all other *humans* and all resources in the environment. Within scenario 2 *humans* have only a limited view of the environment. This scenario could implement a function that the known resources are only handed on to *humans* with the same believe and to all other *humans* with a small probability. In all other cases a *human* “lies” and hands on a (in its view) wrong set of resource locations. This may extend the evaluation if a *human* of a collective belief has any benefit out of the membership.

Further work could contain an analysis of the simulation's reflection on real world situations. If the simulation reflects the behavior of humans in real world the model could help to make forecasts for the behavior of humans in real world situations and to forecast human developments (e.g. urbanization, group's behavior, and religious developments). Furthermore, some key-indicators for collective (mis-) belief can be evaluated. Here a critical mass could exist for distribution of a belief.

Another idea is to implement a learning process into the simulation. An agent compares its *pseudo-agents* (or all agents' location) out of one tick with the agents' location out of the following tick. If there are agents which did not moved the *human* can be sure (by a small correction of an error) that this agent is no real agent. This knowledge could be taken into consideration within its decisions.

All this extensions show that there is much space for further research on the topic of "Simulating Collective Misbelief".

References

Bundesverband, Sekten- und Psychomarktbetrachtung e.V. *Infos über Sekten, Kulte und den Psychomarkt.* 30. 1 2008. http://www.agpf.de/Jonestown.htm.

Davis, J.P., K.M. Eisenhardt, und C.B. Bingham. „Developing theory through simulation methods.“ *The Academy of Management Review*, 2007: 480 - 499.

Dooley, J. *Software Development and Professional Practice.* Apress, 2011.

Doran, Jim. „Simulating Collective Missbelief.“ *Journal of Artificial Societies and Social Simulation*, 1998.

Gilbert, Nigel G., and Rosaria Conte. *Artificial Societies: the Cumputer Simulation of Social Phenomena.* UCL Press, 1995.

Harrison, J.R., Z. Lin, G.R. Carroll, und K.M. Carley. „Simulation modeling in organizational and management research.“ *The Academy of Management Review*, 2007: 1229 - 1245.

Hehlmann, Wilhelm. *Wörterbuch der Psychologie.* Stuttgart: Alfred Körner Verlag, 1965.

Hevner, Alan R., Salvatore T. March, Jinsoo Park, und Sudha Ram. „Design Science in Information Systems Research.“ *MIS Quarterly vol. 28*, 2004: 75-105.

McHaney, R. *Computer Simulation: a practical perspective.* Academic Pr, 1991.

Merten, Klaus. *Einführung in die Kommunikationswissenschaft.* LIT Verlag, 1999.

Olshansky, Brian. „Placebo and nocebo in cardiovascular health: Implications for healthcare, research, and the doctor-patient relationship.“ *Journal of the American College of Cardiology*, 2007.

Railsback, Steven F., Steven L. Lytinen, and Stephen K. Jackson. "Agent-based Simulation Platforms: Review and Development Recommendations." 2006: 609 - 623.

Russell, S., and P. Norvig. *Atrificial Intelligence: A modern approach.* Prentice-Hall, 1995.

Troitzsch, Klaus G., and Nigel Gilbert. *Simulation for the Social Scientist.* Open University Press, 2005.

Wilde, Thomas, and Thomas Hess. "Forschungsmethoden der Wirtschaftsinformatik: Eine empirische Untersuchung." *Wirtschaftsinformatik*, 2007: 280-287.

Wilde, Thomas, and Thomas Hess. *Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung.* München: Institut für Wirtschaftsinformatik und Neue Medien der Ludwig-Maximilians Universität München, 2006.

Wilensky, U. *NetLogo.* 1999. http://ccl.northwestern.edu/netlogo/.

Appendix

Additionally to this thesis the following elements will be published on

http://userpages.uni-koblenz.de/~khemmerich/Abschlussarbeiten/BachelorThesis/

Bachelor Thesis

1. **Thesis**

   1.1. Paper

   The Bachelor Thesis of Kai Hemmerich as .pdf file.

   1.2. Evaluation

   The Excel based evaluation files with some further graphics and the original evaluation databases. A general explanation about what the table is needed for and is used for is located on the top of each table.

   The evaluation of scenario 2 is based on two approaches. The first approach contains the idea that child humans start with an empty friend list in the environment. The only exception is that the child human adds itself as well as its parent to its friend list. Additionally the parent human adds the child to its friend list. This provides the two agents to be a possible target at the next tick. The evaluation results belonging to this approach are marked green (humans can not add further agents to their friend list) and yellow (humans can add further agents to their friend list). The second approach contains the idea that child humans start with the friend list of its parent agent. Additionally the parent human adds the child to its friend list. The evaluation results belonging to this approach are marked light red (humans can not add further agents to their friend list) and light blue (humans can add further agents to their friend list).

2. **Tool**

   2.1. NetLogo4.1.3Installer.exe

   It might be the case that further developments at NetLogo are the reason that the model will not be executable. Therefore the installer-exe will be offered.

   2.2. SimulatingCollectiveMisbelief.nlogo

   The developed implementation of the “Simulatin Collective Misbelief” model (Doran 1998) into the NetLogo language based on NetLogo 4.1.3.